**Pintos Project 0-2: Pintos Data Structure**

|  |  |
| --- | --- |
| 담당 교수 : | 김영재 교수님 |
| 학번 : | 20171487 |
| 이름 : | 최민주 |
|  |  |

**반드시 아래의 양식과 순서를 따라서 작성하기 바랍니다.**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | void parser(); |
| **Parameter** | None |
| **Return** | None |
| **Function** | getCommand() 함수를 호출하여 명령어를 입력받고 반환값의 종류에 따라 다른 함수를 실행시킨다. 명령어는 create, delete, dumpdata, quit, 그리고 자료구조별로 사용하는 others로 분류된 명령어들로 나눈다. |

|  |  |
| --- | --- |
| **Prototype** | FUNCTION getFunction(char\*); |
| **Parameter** | 입력된 명령어 문자열 |
| **Return** | 문자열에 해당하는 인덱스(문자열 종류에 매칭되는 enum 값) |
| **Function** | 입력된 명령어가 정해진 목록 안에 있는지 확인하고 있으면 enum 값을 반환한다. 없다면 -1을 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | COMMAND getCommand(char\*); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | 명령어의 종류에 해당하는 enum 값 |
| **Function** | stdin으로 문자를 입력받아서 입력 값에 따라 명령어 종류를 enum 값에 매칭하여 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool getNameOnly(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | 자료구조의 이름만 인수로 받았는지의 여부 |
| **Function** | stdin으로 한 줄 입력을 받은 후 1개의 인수만 있는지 확인한다. 인수를 한개만 받았다면 해당 인수를 자료구조의 이름이라 간주하고 parameter로 받은 문자열에 저장한 후 true를 반환한다. 1개 이상의 인수를 받았다면 false를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void dumpdata(char\* commandLine); |
| **Parameter** | 이름을 저장할 문자열 |
| **Return** | None |
| **Function** | getNameOnly함수를 통해서 자료구조 이름을 입력받은 후 list, hash, bitmap 목록에서 각각 탐색하는 함수를 호출 후 일치하는 이름이 있다면 해당 자료구조의 원소를 모두 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | void create(char \*commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 생성하고자 하는 자료구조의 종류를 결정하고, 지정된 자료구조가 아니라면 반환. 지정된 자료구조라면 이름을 입력받은 후 해당 자료구조를 생성하는 함수를 각각 호출한다. |

|  |  |
| --- | --- |
| **Prototype** | void delete(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 자료구조의 이름을 입력받은 후 list, hash, bitmap 목록에서 일치하는 이름을 찾아 제거하는 함수를 호출한다. |

|  |  |
| --- | --- |
| **Prototype** | void execute(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | getFunction함수를 통해 명령어의 종류를 결정하고, 미리 설정한 명령 에 없다면 반환, 미리 설정한 명령어 목록에 존재한다면 그에 대응하는 함수를 실행시킨다. |

|  |  |
| --- | --- |
| **Prototype** | void createList(char\* name); |
| **Parameter** | 생성할 리스트의 이름 |
| **Return** | None |
| **Function** | getEmptyList함수를 호출하여 list pool에서 사용하지 않고 있는 리스트를 받아온다. 입력받은 이름을 저장하고, 해당 리스트의 포인터에 메모리를 할당한 후 list\_init 함수를 호출하여 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | int getMatchingList(char\* name); |
| **Parameter** | 리스트 이름 |
| **Return** | 리스트의 인덱스 |
| **Function** | 입력받은 이름을 가진 리스트가 존재하는지 확인하고 존재한다면 pool 내의 인덱스를, 없다면 -1를 반환 |

|  |  |
| --- | --- |
| **Prototype** | int getEmptyList(); |
| **Parameter** | none |
| **Return** | 빈 리스트의 인덱스 |
| **Function** | 리스트 pool에서 할당되지 않은 리스트가 있는지 확인하고 있다면 그 인덱스를 반환 |

|  |  |
| --- | --- |
| **Prototype** | int deleteList(char\* name); |
| **Parameter** | 삭제할 리스트의 이름 |
| **Return** | 삭제한 리스트의 인덱스 |
| **Function** | getMatchingList함수를 통해 해당 이름을 갖는 리스트가 있는지 확인. 있다면 리스트 내의 각 원소를 해제하고 마지막으로 리스트를 해제한다. 마지막에는 삭제한 리스트의 인덱스를 반환한다. 이름과 매치되는 리스트가 없다면 -1를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_front(char\* name); |
| **Parameter** | 리스트의 이름 |
| **Return** | None |
| **Function** | 해당 이름을 갖는 리스트의 인덱스를 받아온 후 list\_front 함수를 리스트에 대해 호출하여 첫 번째 원소를 받아온 후 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_back(char\* name); |
| **Parameter** | 리스트의 이름 |
| **Return** | None |
| **Function** | 해당 이름을 갖는 리스트의 인덱스를 받아온 후 list\_back 함수를 리스트에 대해 호출하여 마지막 원소를 받아온 후 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_push\_front(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 이름과 삽입할 원소로 파싱한 후, 이름을 갖는 리스트가 존재하다면 list\_push\_front함수를 호출하여 해당 리스트의 앞에 해당 값을 넣는다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_push\_back(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 이름과 삽입할 원소로 파싱한 후, 이름을 갖는 리스트가 존재하다면 list\_push\_front함수를 호출하여 해당 리스트의 뒤에 해당 값을 넣는다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_pop\_front(char\* name); |
| **Parameter** | 리스트의 이름 |
| **Return** | None |
| **Function** | 매치되는 이름의 리스트를 찾은 후 list\_pop\_front 함수를 호출하여 해당 리스트의 맨 앞 원소를 제거한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_pop\_back(char\* name); |
| **Parameter** | 리스트의 이름 |
| **Return** | None |
| **Function** | 매치되는 이름의 리스트를 찾은 후 list\_pop\_back 함수를 호출하여 해당 리스트의 맨 뒤 원소를 제거한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_insert(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 이름과 삽입할 위치, 삽입할 원소로 파싱한 후, 이름을 갖는 리스트가 존재하다면 list\_insert 함수를 호출하여 해당 위치에 해당 값을 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_insert\_ordered(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 이름과 삽입할 원소로 파싱한 후, 이름을 갖는 리스트가 존재하다면 list\_insert\_ordered함수를 호출하여 해당 리스트에 순서에 맞도록 원소를 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_empty(char\* name); |
| **Parameter** | 리스트의 이름 |
| **Return** | None |
| **Function** | 매치되는 이름의 리스트를 찾은 후 list\_empty 함수를 호출하여 비었는지 여부를 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_size(char\* name); |
| **Parameter** | 리스트의 이름 |
| **Return** | None |
| **Function** | 매치되는 이름의 리스트를 찾은 후 list\_size 함수를 호출하여 원소의 개수를 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_max(char\* name); |
| **Parameter** | 리스트의 이름 |
| **Return** | None |
| **Function** | 매치되는 이름의 리스트를 찾은 후 list\_max 함수를 호출하여 가장 값이 큰 원소를 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_min(char\* name); |
| **Parameter** | 리스트의 이름 |
| **Return** | None |
| **Function** | 매치되는 이름의 리스트를 찾은 후 list\_min 함수를 호출하여 가장 값이 작은 원소를 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_remove(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 이름과 제거할 원소의 인덱스로 파싱한 후, 이름을 갖는 리스트가 존재하다면 list\_remove 함수를 호출하여 해당 인덱스 위치의 값을 제거한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_reverse(char\* name); |
| **Parameter** | 리스트의 이름 |
| **Return** | None |
| **Function** | 매치되는 이름의 리스트를 찾은 후 list\_reverse를 호출하여 원소 순서를 반대로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_sort(char\* name); |
| **Parameter** | 리스트의 이름 |
| **Return** | None |
| **Function** | 매치되는 이름의 리스트를 찾은 후 list\_sort 함수를 호출하여 원소의 데이터 순서대로 정렬한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_splice(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 삽입할 리스트의 이름과 위치, splice할 리스트의 이름과 처음 위치, 마지막 위치로 파싱한 후 매치되는 이름의 리스트가 각각 존재할 경우 list\_splice 함수를 호출하여 범위의 원소를 다른 리스트에 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_swap(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 리스트의 이름과 두 개의 인덱스로 파싱한 후 list\_swap함수를 호출하여 두 원소의 위치를 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_unique(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 두 개의 리스트의 이름으로 파싱한 후 매치되는 이름이 있는지 확인한 후 list\_unique 함수를 실행하여 중복되는 원소 없이 리스트를 복제한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_list\_shuffle(char\* name); |
| **Parameter** | 리스트의 이름 |
| **Return** | None |
| **Function** | 매치되는 이름의 리스트를 찾은 후 list\_shuffle을 수행하여 원소의 순서를 무작위로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | int getMatchingHash(char\* name); |
| **Parameter** | 해시 테이블의 이름 |
| **Return** | 해시의 인덱스 |
| **Function** | 입력받은 이름을 가진 해시 테이블이 존재하는지 확인하고 존재한다면 pool 내의 인덱스를, 없다면 -1를 반환 |

|  |  |
| --- | --- |
| **Prototype** | int getEmptyHash(); |
| **Parameter** | None |
| **Return** | 비어 있는 해시의 인덱스 |
| **Function** | 해시 pool 내에서 비어있는 위치를 찾아서 인덱스를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void createHash(char\* name); |
| **Parameter** | 생성할 해시 테이블의 이름 |
| **Return** | None |
| **Function** | GetEmptyHash함수를 호출하여 hash pool에서 사용하지 않고 있는 리스트를 받아온다. 입력받은 이름을 저장하고, 해당 리스트의 포인터에 메모리를 할당한 후 hash\_init 함수를 호출하여 초기화한다. |

|  |  |
| --- | --- |
| **Prototype** | int deleteHash(char\* name); |
| **Parameter** | 삭제할 해시 테이블의 이름 |
| **Return** | 삭제된 해시 테이블의 이름 |
| **Function** | getMatchingHash함수를 통해 해당 이름을 갖는 해시가 있는지 확인. 있다면 hash\_delete 함수를 실행 후 해시 내의 각 원소를 해제하고 마지막으로 해시를 해제한다. 마지막에는 삭제한 해시의 인덱스를 반환한다. 이름과 매치되는 리스트가 없다면 -1를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_hash\_insert(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 해시의 이름과 입력값으로 파싱한 후 매치되는 이름이 있는지 확인하고 hash\_insert를 통해 해시 원소를 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_hash\_apply(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 해시의 이름과 액션 함수로 파싱한 후 매치되는 이름이 있는지 확인하고 hash\_apply를 통해 함수를 모든 원소에 적용시킨다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_hash\_delete(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 해시 이름과 제거할 값으로 파싱하고 매치되는 이름이 있는지 확인한 후 hash\_delete를 통해 해당 값을 갖는 해시 원소를 제거한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_hash\_empty(char\* name); |
| **Parameter** | 해시 테이블의 이름 |
| **Return** | None |
| **Function** | 매치되는 해시 테이블의 이름이 있는지 확인 후 해당 해시가 비었는지 hash\_empty 함수를 이용하여 확인. |

|  |  |
| --- | --- |
| **Prototype** | void func\_hash\_size(char\* name); |
| **Parameter** | 해시 테이블의 이름 |
| **Return** | None |
| **Function** | 매치되는 해시 테이블의 이름이 있는지 확인 후 해당 해시 테이블의 크기를 hash\_size 함수를 이용하여 확인 |

|  |  |
| --- | --- |
| **Prototype** | void func\_hash\_clear(char\* name); |
| **Parameter** | 해시 테이블의 이름 |
| **Return** | None |
| **Function** | 매치되는 해시 테이블의 이름이 있는지 확인 후 해당 해시 테이블의 원소를 모두 제거한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_hash\_find(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 해시의 이름과 찾으려는 값으로 파싱하고 매치되는 이름이 있는지 확인한 후 hash\_find 함수를 호출하여 찾은 원소를 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_hash\_replace(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 해시의 이름과 삽입하려는 값으로 파싱하고 매치되는 이름이 있는지 확인한 후 hash\_replace 함수를 호출하여 값을 삽입하는데, 이미 같은 값이 존재하다면 삭제하고 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | int getMatchingBitmap(char\* name); |
| **Parameter** | 비트맵의 이름 |
| **Return** | 비트맵의 인덱스 |
| **Function** | 입력받은 이름을 가진 비트맵이 존재하는지 확인하고 존재한다면 pool 내의 인덱스를, 없다면 -1를 반환 |

|  |  |
| --- | --- |
| **Prototype** | int getEmptyBitmap(); |
| **Parameter** | None |
| **Return** | 비트맵의 인덱스 |
| **Function** | 비어 있는 비트맵이 존재하는지 확인하고 존재한다면 pool 내의 인덱스를, 없다면 -1을 반환 |

|  |  |
| --- | --- |
| **Prototype** | void createBitmap(char\* name, int bitmap\_cnt); |
| **Parameter** | 비트맵의 이름과 생성할 비트맵의 크기 |
| **Return** | None |
| **Function** | 비어있는 비트맵의 인덱스를 받아 해당 위치에 bitmap\_create를 이용하여 bitmap\_cnt 개의 비트를 갖는 비트맵을 생성한다. |

|  |  |
| --- | --- |
| **Prototype** | int deleteBitmap(char\* name); |
| **Parameter** | 비트맵의 이름 |
| **Return** | 제거된 비트맵의 인덱스 |
| **Function** | 비트맵의 이름을 받아 매칭되는 비트맵이 있다면 bitmap\_destroy를 이용하여 제거한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_mark(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 비트맵의 이름과 인덱스로 파싱하고 매치되는 이름이 있는지 확인한 후 bitmap\_mark 함수를 호출하여 해당 index에 true 값을 입력한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_all(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 비트맵의 이름과 시작 인덱스와 개수로 파싱하고 매치되는 이름이 있는지 확인한 후 bitmap\_all 함수를 호출하여 시작 인덱스부터 cnt 개수까지의 비트를 모두 true로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_any(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 비트맵의 이름과 시작 인덱스, 개수로 파싱하고 매치되는 이름이 있는지 확인한 후 bitmap\_any 함수를 호출하여 시작 인덱스부터 cnt 개수까지의 범위에 true가 하나라도 존재하는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_contains(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 비트맵의 이름과 시작 인덱스, 셀 개수로 파싱하고 매치되는 이름이 있는지 확인한 후 bitmap\_contains 함수를 호출하여 start부터 cnt개수의 비트 내에 해당 값이 존재하는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_count(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 비트맵의 이름과 시작 인덱스, 비트 개수로 파싱하고 매치되는 이름이 있는지 확인한 후 bitmap\_count 함수를 호출하여 start 부터 cnt 개수의 비트 내에 해당 값을 갖는 비트의 수를 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_expand(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 비트맵의 이름과 늘릴 개수로 파싱하고 매치되는 이름이 있는지 확인한 후 bitmap\_expand 함수를 호출하여 현재 비트맵에서 cnt 개수만큼 비트를 늘린다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_set\_all(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 비트맵의 이름과 설정할 값으로 파싱하고 매치되는 이름이 있는지 확인한 후 bitmap\_set\_all 함수를 호출하여 입력된 값으로 모든 비트를 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_flip(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 비트맵의 이름과 인덱스로 파싱하고 매치되는 이름이 있는지 확인한 후 bitmap\_flip 함수를 호출하여 비트맵의 해당 인덱스 값을 반전시킨다 |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_none(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 비트맵의 이름과 시작 인덱스, 비트 개수로 파싱하고 매치되는 이름이 있는지 확인한 후 bitmap\_none 함수를 호출하여 start 인덱스부터 cnt 개의 비트가 모두 false인지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_reset(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 비트맵의 이름과 인덱스로 파싱하고 매치되는 이름이 있는지 확인한 후 bitmap\_reset 함수를 호출하여 해당 인덱스 값을 false로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_scan\_and\_flip(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 비트맵의 이름과 시작 인덱스, 비트 개수, 비트값으로 파싱하고 매치되는 이름이 있는지 확인한 후 bitmap\_scan\_and\_flip 함수를 호출하여 start 인덱스부터 cnt 개수의 비트 중 입력된 비트값과 같은 값을 같은 첫 번째 인덱스를 찾고 해당 비트부터 다시 cnt 개수의 비트만큼 모든 값을 반전시킨다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_scan(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 비트맵의 이름과 시작 인덱스, 비트 개수, 비트값으로 파싱하고 매치되는 이름이 있는지 확인한 후 bitmap\_scan\_and\_flip 함수를 호출하여 start 인덱스부터 cnt 개수의 비트 중 입력된 비트값과 같은 값을 같은 첫 번째 인덱스를 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_set(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 비트맵의 이름과 인덱스, 비트값으로 파싱하고 매치되는 이름이 있는지 확인한 후 bitmap\_set 함수를 호출하여 해당 비트맵의 인덱스 값을 입력된 비트값으로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_set\_multiple(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 비트맵의 이름과 시작 인덱스, 비트 개수, 비트값으로 파싱하고 매치되는 이름이 있는지 확인한 후 bitmap\_set\_multiple 함수를 호출하여 start 인덱스부터 cnt 개수만큼 입력된 비트값으로 세팅한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_test(char\* commandLine); |
| **Parameter** | 입력값을 저장할 문자열 |
| **Return** | None |
| **Function** | 한 줄 입력을 받아 비트맵의 이름과 인덱스로 파싱하고 매치되는 이름이 있는지 확인한 후 bitmap\_test 함수를 호출하여 해당 인덱스의 값을 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_dump(char\* name); |
| **Parameter** | 비트맵의 이름 |
| **Return** | None |
| **Function** | 매칭되는 비트맵의 이름이 있는지 확인한 후 bitmap\_dump 함수를 호출하여 해당 비트맵의 hex 정보를 출력한다. |

|  |  |
| --- | --- |
| **Prototype** | void func\_bitmap\_size(char\* name); |
| **Parameter** | 비트맵의 이름 |
| **Return** | None |
| **Function** | 매칭되는 비트맵의 이름이 있는지 확인한 후 bitmap\_size 함수를 호출하여 해당 비트맵의 크기를 출력한다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_head (struct list\_elem \*elem); |
| **Parameter** | Head node인지 확인할 list element |
| **Return** | Head node의 여부 |
| **Function** | 해당 element가 head node인지 확인 |

|  |  |
| --- | --- |
| **Prototype** | static bool is\_sorted (struct list\_elem \*a, struct list\_elem \*b,  list\_less\_func \*less, void \*aux); |
| **Parameter** | 순서를 확인할 두개의 list element, 대소 관계 확인하는 함수 |
| **Return** | 순서대로 배치되어 있는지의 여부 |
| **Function** | 리스트의 두 노드를 받아 순서대로 배치되었는지 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_interior (struct list\_elem \*elem); |
| **Parameter** | 내부 node인지 확인할 list element |
| **Return** | 내부 node인지 여부 |
| **Function** | 해당 element가 list 내부node인지 확인 |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_tail (struct list\_elem \*elem); |
| **Parameter** | Tail node인지 확인할 list element |
| **Return** | Tail node인지의 여부 |
| **Function** | 해당 element가 tail node인지 확인 |

|  |  |
| --- | --- |
| **Prototype** | void list\_init (struct list \*); |
| **Parameter** | 초기화 할 리스트를 새로 할당해서 받음 |
| **Return** | None |
| **Function** | 새로 생성한 리스트를 받아서 head node와 tail node를 초기화해준다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_begin (struct list \*); |
| **Parameter** | 첫번째 원소를 확인할 리스트 |
| **Return** | 첫번째 리스트 원소 |
| **Function** | 리스트를 받아서 그 리스트의 첫번째 원소 node를 반환해준다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next (struct list\_elem \*); |
| **Parameter** | 다음 원소를 확인할 리스트 원소 |
| **Return** | 입력된 원소의 다음 원소 |
| **Function** | node element를 받아서 연결된 다음 node를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_end (struct list \*); |
| **Parameter** | Tail node를 확인할 리스트 |
| **Return** | Tail node |
| **Function** | 리스트를 받아서 그 리스트의 tail node를 반환해준다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rbegin (struct list \*); |
| **Parameter** | 마지막 원소를 얻을 리스트 |
| **Return** | 마지막 원소 node |
| **Function** | 리스트를 받아서 그 리스트의 마지막 원소 node를 반환해준다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_prev (struct list\_elem \*); |
| **Parameter** | 앞의 원소를 확인할 리스트 원소 |
| **Return** | 입력된 원소의 앞에 위치한 원소 |
| **Function** | node element를 받아서 연결된 이전 node를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rend (struct list \*); |
| **Parameter** | Head node를 얻을 리스트 |
| **Return** | 리스트의 head node |
| **Function** | 리스트를 받아서 그 리스트의 head node를 반환해준다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head (struct list \*); |
| **Parameter** | Head node를 얻을 리스트 |
| **Return** | 리스트의 head node |
| **Function** | 리스트를 받아서 그 리스트의 head node를 반환해준다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_tail (struct list \*); |
| **Parameter** | Tail node를 얻을 리스트 |
| **Return** | 리스트의 tail node |
| **Function** | 리스트를 받아서 그 리스트의 tail node를 반환해준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert (struct list\_elem \*, struct list\_elem \*); |
| **Parameter** | 새로 넣을 리스트 원소와, 삽입할 위치 |
| **Return** | None |
| **Function** | 새로운 노드를 받아서 입력된 노드 앞쪽에 넣어준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice (struct list\_elem \*before,  struct list\_elem \*first, struct list\_elem \*last); |
| **Parameter** | 이동할 원소의 범위를 나타내는 element(첫번째 원소부터 마지막 원소 다음의 원소), 원소 범위를 삽입할 위치를 나타내는 element |
| **Return** | None |
| **Function** | “first”부터 “last” 이전의 node 들을 before node앞으로 옮겨준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front (struct list \*, struct list\_elem \*); |
| **Parameter** | 삽입할 리스트와 삽입할 원소 |
| **Return** | None |
| **Function** | 리스트의 가장 앞 노드에 새로운 노드를 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back (struct list \*, struct list\_elem \*); |
| **Parameter** | 삽입할 리스트와 삽입할 원소 |
| **Return** | None |
| **Function** | 리스트의 가장 마지막 노드에 새로운 노드를 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove (struct list\_elem \*); |
| **Parameter** | 제거할 원소 |
| **Return** | 제거된 원소 |
| **Function** | 해당 노드를 리스트에서 제거한 뒤 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \*); |
| **Parameter** | 앞의 원소를 제거할 리스트 |
| **Return** | 제거된 원소 |
| **Function** | 리스트의 가장 앞 노드를 리스트에서 제거한 뒤 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \*); |
| **Parameter** | 뒤의 원소를 제거할 리스트 |
| **Return** | 제거된 원소 |
| **Function** | 리스트의 가장 마지막 노드를 리스트에서 제거한 뒤 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front (struct list \*); |
| **Parameter** | 앞의 원소를 얻으려는 리스트 |
| **Return** | 리스트 맨 앞의 원소 |
| **Function** | 리스트를 받아서 가장 앞 노드를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back (struct list \*); |
| **Parameter** | 뒤의 원소를 얻으려는 리스트 |
| **Return** | 리스트 맨 의 원소 |
| **Function** | 리스트의 가장 마지막 노드를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*); |
| **Parameter** | 크기를 얻으려는 리스트 |
| **Return** | 리스트의 원소 개수 |
| **Function** | 리스트의 원소 개수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list \*); |
| **Parameter** | 비었는지 확인할 리스트 |
| **Return** | 리스트의 empty여부 |
| **Function** | 리스트가 비었는지 여부를 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse (struct list \*); |
| **Parameter** | 순서를 바꿀 리스트 |
| **Return** | None |
| **Function** | 리스트의 순서를 반대로 바꿔준다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort (struct list \*,  list\_less\_func \*, void \*aux); |
| **Parameter** | 정렬할 리스트, 대소를 확인하는 함수, 추가정보 |
| **Return** | None |
| **Function** | 리스트를 받아서 원소를 크기 순서대로 정렬한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered (struct list \*,  struct list\_elem \*, list\_less\_func \*, void \*aux); |
| **Parameter** | 이미 정렬된 리스트, 새로 넣을 원소, 대소 관계 함수, 추가 정보 |
| **Return** | None |
| **Function** | 정렬된 리스트에 새로운 원소를 순서에 맞추어 삽입한다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique (struct list \*, struct list \*duplicates,  list\_less\_func \*, void \*aux); |
| **Parameter** | 원본 리스트, 원소를 복제하여 받을 리스트, 대소 관계 함수, 추가 정보 |
| **Return** | None |
| **Function** | 리스트의 모든 원소들을 중복되지 않게 새로운 리스트로 넣는다. |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max (struct list \*, list\_less\_func \*, void \*aux); |
| **Parameter** | 최댓값을 찾을 리스트, 대소 관계 함수, 추가 정보 |
| **Return** | 리스트 내의 최댓값 노드 |
| **Function** | 리스트에서 가장 값이 큰 노드를 찾아서 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min (struct list \*, list\_less\_func \*, void \*aux); |
| **Parameter** | 최솟값을 찾을 리스트, 대소 관계 함수, 추가 정보 |
| **Return** | 리스트 내의 최솟값 노드 |
| **Function** | 리스트에서 가장 값이 작은 노드를 찾아서 반환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem \*a, struct list\_elem \*b); |
| **Parameter** | 위치를 바꿀 두 개의 노드 |
| **Return** | None |
| **Function** | 두 개의 노드를 받아서 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list); |
| **Parameter** | 순서를 무작위로 바꿀 리스트 |
| **Return** | None |
| **Function** | 리스트 내의 원소 순서를 무작위로 바꾼다. |

|  |  |
| --- | --- |
| **Prototype** | bool my\_list\_less\_func(const struct list\_elem \*elem, const struct list\_elem\* e, void \*aux); |
| **Parameter** | 대소를 비교할 두 개의 원소, 추가 정보 |
| **Return** | elem이 e보다 작은지의 여부 |
| **Function** | 두 개의 노드를 받아서 크기를 비교하여 대소 관계를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | #define list\_entry(LIST\_ELEM, STRUCT, MEMBER) \  ((STRUCT \*) ((uint8\_t \*) &(LIST\_ELEM)->next \  - offsetof (STRUCT, MEMBER.next))) |
| **Parameter** | 내부의 struct 주소, 감싸고 있는 struct, 내부의 struct 멤버변수 이 |
| **Return** | 감싸고 있는 struct의 주소 |
| **Function** | 내부의 struct 주소를 받아 그 struct를 포함하고 있는 외부의 struct 주소를 반환한다. |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | #define hash\_entry(HASH\_ELEM, STRUCT, MEMBER) \  ((STRUCT \*) ((uint8\_t \*) &(HASH\_ELEM)->list\_elem \  - offsetof (STRUCT, MEMBER.list\_elem))) |
| **Parameter** | 해쉬 원소의 주소, 해쉬 원소를 감싸고 있는 구조체 종류, 해쉬 원소의 멤버 변수 이름 |
| **Return** | 해쉬 원소를 감싸고 있는 구조체의 주소 |
| **Function** | 해쉬 원소를 받아서 해당 원소를 감싸고 있는 구조체 주소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | #define list\_elem\_to\_hash\_elem(LIST\_ELEM) \  list\_entry(LIST\_ELEM, struct hash\_elem, list\_elem) |
| **Parameter** | 리스트 원소 |
| **Return** | 리스트 원소를 감싸고 있는 해쉬 원소의 주소 |
| **Function** | 리스트 원소를 받아서 해당 원소를 감싸고 있는 해쉬 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init (struct hash \*, hash\_hash\_func \*, hash\_less\_func \*, void \*aux); |
| **Parameter** | 새로 할당한 해시 구조체의 주소, 해시 함수로 사용할 함수, 해시 원소의 대소를 비교할 함수, 추가 정보 |
| **Return** | 초기화가 잘 됬는지 여부 |
| **Function** | 각 멤버 변수의 값과 메모리를 할당하며 해시 테이블을 초기화 한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \*, hash\_action\_func \*); |
| **Parameter** | 비울 해시 구조체의 주소, 원소에 적용할 함수 |
| **Return** | None |
| **Function** | 해시 테이블의 모든 bucket의 원소를 비운다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \*, hash\_action\_func \*); |
| **Parameter** | 제거할 해시 구조체의 주소, 원소에 적용할 함수 |
| **Return** | None |
| **Function** | 해시 테이블의 원소를 모두 지우고, 해당 해시 테이블도 메모리 해제한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert (struct hash \*, struct hash\_elem \*); |
| **Parameter** | 원소를 삽입할 해시의 주소, 삽입할 해시 원소 |
| **Return** | 삽입된 해시 원소의 주소 |
| **Function** | 새로운 해시 원소를 받아서 해시 테이블에 넣고, 해당 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace (struct hash \*, struct hash\_elem \*); |
| **Parameter** | 원소가 삽입될 해시 구조체의 주소, 삽입할 해시 원소 |
| **Return** | 삽입된 해시 원소의 주소 |
| **Function** | 해시에 새로운 해시 원소를 넣되, 이미 존재하다면 해당 원소를 제거한 후 그 자리를 대체한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find (struct hash \*, struct hash\_elem \*); |
| **Parameter** | 값을 찾을 해시 구조체, 찾는 값을 가진 해시 원소 |
| **Return** | 찾는 값을 가진 해시 원소의 주소 |
| **Function** | 입력된 해시 원소와 같은 값을 갖는 원소가 해시 테이블에 존재하는지 확인하고 존재한다면 그 원소의 위치를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete (struct hash \*, struct hash\_elem \*); |
| **Parameter** | 값을 삭제할 해시 구조체, 삭제할 값을 가진 해시 원소 |
| **Return** | 삭제된 해시 원소 |
| **Function** | 입력된 해시 원소와 같은 값을 갖는 원소가 해시 테이블에 존재하는지 확인하고 존재한다면 그 원소를 해시 테이블에서 삭제하고 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \*, hash\_action\_func \*); |
| **Parameter** | 함수를 적용할 해시 구조체, 각 원소에 적용할 함수 |
| **Return** | None |
| **Function** | 해시 테이블의 모든 원소에 함수를 적용한다. |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first (struct hash\_iterator \*, struct hash \*); |
| **Parameter** | 원소의 위치를 받아올 이터레이터, 원소의 위치를 알고자 하는 해시 구조체 |
| **Return** | None |
| **Function** | 이터레이터를 받아서 해시 테이블의 가장 첫 번째 원소로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next (struct hash\_iterator \*); |
| **Parameter** | 포인터를 변경할 이터레이터 |
| **Return** | 변경된 포인터가 가르키는 해시 원소의 주소 |
| **Function** | 이터레이터가 현재 가르키고 있는 원소의 다음 원소를 가르키도록 한다. |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_cur (struct hash\_iterator \*); |
| **Parameter** | 현재 원소를 알고자 하는 이터레이터 |
| **Return** | 이터레이터가 가르키고 있는 해시 원소 |
| **Function** | 현재 이터레이터가 가르키고 있는 해시 원소를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*); |
| **Parameter** | 크기를 알고자 하는 해시 구조체 |
| **Return** | 해시 테이블 내의 원소 개수 |
| **Function** | 현재 해시 테이블의 원소 개수를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \*); |
| **Parameter** | 비었는지 알고자 하는 해시 구조체 |
| **Return** | 해시 테이블이 비었는지의 여부 |
| **Function** | 현재 해시 테이블이 비었는지 여부를 반환한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_string (const char \*); |
| **Parameter** | 키 값을 얻고자 하는 문자열 |
| **Return** | 문자열에 대한 해시 함수의 키 |
| **Function** | 문자열에 대한 해시 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int (int); |
| **Parameter** | 키 값을 얻고자 하는 정수형 |
| **Return** | 정수형 값에 대한 해시 함수의 |
| **Function** | 정수형 값에 대한 해시 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned my\_hash\_func(const struct hash\_elem\* e, void\* aux); |
| **Parameter** | 해시 키 값을 얻고자 하는 해시 원소, 추가 정보 |
| **Return** | 해당 원소의 데이터에 대한 해시 함수의 키 값 |
| **Function** | 정수형 값에 대한 해시 함수를 포함하여 해시 원소에 대해 수행할 수 있게 해주는 함수이다. |

|  |  |
| --- | --- |
| **Prototype** | bool my\_hash\_less\_func (const struct hash\_elem \*a,  const struct hash\_elem \*b,  void \*aux); |
| **Parameter** | 크기 대소를 확인할 두 해시 원소 |
| **Return** | A 원소가 b 원소보다 작은지의 여부 |
| **Function** | 두 개의 해시 원소를 받아서 크기 대소를 확인한다. |

|  |  |
| --- | --- |
| **Prototype** | void my\_hash\_action\_func(struct hash\_elem \*e, void \*aux); |
| **Parameter** | 함수를 적용할 해시 원소 |
| **Return** | None |
| **Function** | 해시 원소를 받아서 해당 원소의 데이터에 action 함수를 적용한다. |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2(int i); |
| **Parameter** | 정수형 값 |
| **Return** | 입력된 정수 값에 대한 해시 함수의 키 |
| **Function** | 직접 만든 정수형 값에 대한 해시 함수이다.  (root(5) – 1) / 2 = A라는 값을 가지고 (i \* A % 1) \* 4라는 연산을 한다. 4는 해시 테이블의 초기 bucket size이다. |

|  |  |
| --- | --- |
| **Prototype** | void square\_func (struct hash\_elem \*e, void \*aux); |
| **Parameter** | 함수를 적용할 해시 원소, 추가 정보 |
| **Return** | None |
| **Function** | action function으로 사용하는 함수로, 해시 원소를 받아서 해당 데이터에 제곱을 적용한다. |

|  |  |
| --- | --- |
| **Prototype** | void triple\_func (struct hash\_elem \*e, void \*aux); |
| **Parameter** | 함수를 적용할 해시 원소, 추가 정보 |
| **Return** | None |
| **Function** | action function으로 사용하는 함수로, 해시 원소를 받아서 해당 데이터에 세제곱을 적용한다. |

|  |  |
| --- | --- |
| **Prototype** | void dealloc\_func (struct hash\_elem \*e, void \*aux); |
| **Parameter** | 해제할 해시 원소, 추가 정보 |
| **Return** | None |
| **Function** | destructor function 으로 사용하며, 해시 원소를 해시 테이블에서 삭제하고 메모리 해제할 때 사용한다. 해시 원소를 감싸고 있는 해시 아이템을 메모리 해제한다. |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create (size\_t bit\_cnt); |
| **Parameter** | 원하는 비트의 사이즈 |
| **Return** | 생성한 비트맵 |
| **Function** | 비트 개수를 받아서 해당 크기의 비트맵을 생성한다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create\_in\_buf (size\_t bit\_cnt, void \* block, size\_t byte\_cnt); |
| **Parameter** | 비트 크기, 블락의 주소, 블락의 크기 |
| **Return** | 비트맵 |
| **Function** | Creates and returns a bitmap with BIT\_CNT bits in the BLOCK\_SIZE bytes of storage preallocated at BLOCK. BLOCK\_SIZE must be at least bitmap\_needed\_bytes(BIT\_CNT). |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_buf\_size (size\_t bit\_cnt); |
| **Parameter** | 바이트 크기 |
| **Return** | 비트 카운트 |
| **Function** | bitmap\_create\_in\_buf를 사용하기 위해 필요한 바이트의 수를 알려준다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*); |
| **Parameter** | 없애려는 비트맵 |
| **Return** | None |
| **Function** | 들어온 비트맵의 멤버 변수들을 해제하고 해당 비트맵도 해제한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*); |
| **Parameter** | 크기를 알려고 하는 비트맵 |
| **Return** | 비트맵의 크기 |
| **Function** | 들어온 비트맵의 비트 사이즈를 알려준다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set (struct bitmap \*, size\_t idx, bool); |
| **Parameter** | 설정하고자 하는 비트맵, 설정하고자 하는 비트의 인덱스, 설정할 값 |
| **Return** | None |
| **Function** | 비트맵의 idx 번째 비트에 bool 값을 입힌다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \*, size\_t idx); |
| **Parameter** | mark할 bitmap, mark할 인덱스 |
| **Return** | None |
| **Function** | bitmap의 idx번째 비트를 true로 만든다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset (struct bitmap \*, size\_t idx); |
| **Parameter** | reset할 bitmap, reset할 인덱스 |
| **Return** | None |
| **Function** | bitmap의 idx번째 비트를 false로 만든다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \*, size\_t idx); |
| **Parameter** | 값을 반전할 bitmap, 반전할 인덱스 |
| **Return** | None |
| **Function** | bitmap의 idx번째 비트를 반전시킨다. |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test (const struct bitmap \*, size\_t idx); |
| **Parameter** | 값을 확인할 bitmap, 값을 확인할 idx |
| **Return** | bool |
| **Function** | bitmap의 idx번째 인덱스 값이 무엇인지 확인 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \*, bool); |
| **Parameter** | set할 bitmap, set할 값 |
| **Return** | None |
| **Function** | bitmap의 모든 비트에 입력된 bool 값을 적용시킨다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | 값을 설정할 비트맵, 값을 설정하기 시작할 인덱스, 값을 설정할 비트 개수, 설정할 값 |
| **Return** | None |
| **Function** | bitmap의 start번째 비트부터 cnt 개의 비트를 bool 값으로 설정한다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | 카운트 할 bitmap, 카운트를 시작할 인덱스, 카운트를 할 개수, 카운트 대상이 될 값 |
| **Return** | 값을 가진 비트의 수 |
| **Function** | bitmap의 start부터 cnt개의 비트 안에 bool 값이 몇개 있는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | 값을 가지고 있는지 확인할 bitmap, 확인을 시작할 인덱스, 확인할 비트의 개수, 확인할 값 |
| **Return** | 값을 가지고 있는지의 여부 |
| **Function** | bitmap의 start부터 cnt개의 비트 안에 bool 값이 하나라도 있는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | 확인할 bitmap, 확인을 시작할 인덱스, 확인할 비트의 수 |
| **Return** | true가 있는지의 여부 |
| **Function** | bitmap의 start부터 start+cnt – 1 까지 중에 true가 하나라도 있는지 확인 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | 확인할 bitmap, 확인을 시작할 인덱스, 확인할 비트의 수 |
| **Return** | true가 없는지의 여부 |
| **Function** | bitmap의 start부터 start+cnt – 1 까지 중에 모두 false인지 확인 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | 확인할 bitmap, 확인을 시작할 인덱스, 확인할 비트의 수 |
| **Return** | 모두 true인지의 여부 |
| **Function** | bitmap의 start부터 start+cnt – 1 까지 중에 모두 true인지 확인 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | 확인할 bitmap, 확인을 시작할 인덱스, 확인할 비트의 수, 확인할 값 |
| **Return** | 비트의 인덱스 |
| **Function** | bitmap의 start부터 start+cnt – 1 까지 확인하며 확인할 값이 있는 첫번째 인덱스를 찾는다. |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | 확인할 bitmap, 확인을 시작할 인덱스, 확인할 비트의 수, 확인할 값 |
| **Return** | 반전시킨 비트의 첫 인덱스 |
| **Function** | bitmap의 start부터 start+cnt – 1 까지 입력 값에 대해 scan하며 찾는다면 해당 위치부터 cnt개의 비트를 반전시킨다. |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump (const struct bitmap \*); |
| **Parameter** | 출력할 비트맵 |
| **Return** | None |
| **Function** | 비트맵을 입력받아 hex\_dump를 실행시킨다. |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap\* bitmap\_expand(struct bitmap \*old, int new\_cnt); |
| **Parameter** | 기존의 비트맵, 연장할 비트의 개수 |
| **Return** | 새로 생성한 비트맵 |
| **Function** | 기존의 비트맵에서 입력된 개수의 비트를 연장한다. |